

JEGYZŐKÖNYV

Operációs Rendszerek BSc

2022. Tavasz féléves feladat

Készítette: Czikó Tivadar

Neptun kód: O2IXLB

A feladat leírása:

Írjon egy C nyelvű programokat, ami:

Létrehoz egy üzenetsort, ebben az üzenetsorba betesz másodpercenként egy változót, ennek a változónak a kezdőértéke legyen az aktuális processz azonosítója a változó mindig beírás után növekedjen 2-vel. A másik program pedig: kiolvas.

A feladat elkészítésének lépései:

A feladat elkészítéséhez kettő c programra és egy header állományra lesz szükség. Az egy üzenetet küld a másik pedig kiolvassa, a végén pedig eltávolítja az üzenetsort. A header fájlban egy struktúrát definiáltam az üzenet létrehozásához, illetve az üzenet számát és hosszát is ebben határoztam meg.

Először a küldő programmal kezdem (msgsender.c). Első lépés egy kulcsot generálunk a queue id létrehozásához (queue.h). A 'queue.h' header állományra azért van szükség mert, ha egy szabadon választott számot használunk akkor van rá lehetőség, hogy összeütközésbe kerül más, nem ezzel problémával foglalkozó programmal.

Header állomány (queue.h):

```
#define ProjectId 123
#define PathName "queue.h"
#define MsgLen 50
#define MsgCount 10

typedef struct {
    int id;
    int value;
} queuedMessage;
```

Második lépésként létrehozunk egy üzenetsort, ahol a 0666 kapcsoló a hozzáférésért felelős a memóriaszegmenshez, míg az IPC_CREAT utasítást ad a rendszernek, hogy készítsen egy új memóriaszegmenst az osztott memóriának.

Következőképpen egy 'for' cikluson belül létrehozom a küldésre kész üzenetet. Itt a ciklus előre megadott értékig megy, azt végtelen ciklusra cserélve a program futásának leállításig történik. A küldendő üzenet a header fájlban létrehozásra került struktúra, ahol a value az aktuális processz értékével kezdődik, az id pedig a küldött üzenet sorszáma.

Ez után létrehozásra került a küldendő üzenet, az 'msgsnd' használatával küldöm, ahol az IPC_NOWAIT kapcsolót használtam arra az esetre, ha tele lenne az üzenetsor, akkor ne vározzon tovább.

Üzenet küldő c program kódja (msgsender.c):

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdlib.h>
#include <string.h>
```

```

#include <unistd.h>
#include "queue.h"

void report_and_exit(const char* msg) {
    perror(msg);
    exit(-1); /* Hiba eseten killep */
}

int main() {
    int value;
    key_t key = ftok(PathName, ProjectId); /* kulcs generalasa*/
    if (key < 0) report_and_exit("Nem kaphato kulcsot!");

    int qid = msgget(key, 0666 | IPC_CREAT); /* generalt kulcs hasznalata queue
id kereshez*/
    if (qid < 0) report_and_exit("Nem kaphato queue id!");
    value = qid;
    int i;
    for (i = 0; i < MsgCount; i++) {
        /* uzenetsor generalasa, queuedMessage a queue.h fileban definialt
struktura */
        queuedMessage msg;
        msg.value = value;
        msg.id = i;

        /* uzenet kuldes */
        msgsnd(qid, &msg, sizeof(msg), IPC_NOWAIT); /* ha az uzenetsor tele, nem
ir uzenetet, hanem visszater a processz elejere */
        printf("%i Uzenet elkuldve! Ertek: %i\n", (int) msg.id, (int) msg.value);
        value = value + 2;
        sleep(1);
    }
    return 0;
}

```

Miután elküldtem az üzenetet, kiírom az elküldött üzenet tartalmát és értékét, majd növelem a változó értékét 2-vel és a program vár 1 másodpercet, ezután küldi a következő üzenetet.

Harmadik lépésként szükség van az üzenetek fogadására szolgáló programra. Ebben az esetben is szükségünk lesz kulcs generálásra, hogy azonosítani tudjuk az üzenetsort.

A fogadó program nem készít üzenetsort, amiatt megtévesztő lehet az IPC_CREAT kapcsoló, de ez azt jelent ebben az esetben, hogy ha szükséges elkészíti, egyébként csak hozzáfér.

Míg a küldő programnál az 'msgsnd'-et használtuk addig, itt a 'msgrcv'-et használtuk a fogadásra. itt a 0 kapcsoló az üzenetek fogadását, érkezési sorrendben hajtja végre, az MSG_NOERROR kapcsoló arra szolgál, hogy ha a kapott adat mérete nagyobb mint a megadott érték (msgsz), ne térjen vissza hibával, az IPC_NOWAIT pedig azért, hogy ha nem érhető el az üzenet, ne várjon rá, hanem térjen vissza azonnal.

A(z) `'msgreceiver.c'` futása hasonló a(z) `'msgsender.c'` programéhoz, hogy itt is kiírja az üzenetben kapott értéket, majd 1 másodpercet vár és utána a következő üzenet fog érkezni.

A fogadó program végén, miután megkapta az összes értéket, szükséges eltávolítani az üzenetsort az `'msgctl'` hívással.

Az fogadó c program kódja (`msgreceiver.c`):

```
#include <stdio.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdlib.h>
#include "queue.h"

void report_and_exit(const char* msg) {
    perror(msg);
    exit(-1); /* hiba eseten kilep */
}

int main() {
    key_t key= ftok(PathName, ProjectId); /* kulcs az uzenetsor azonositasara */
    if (key < 0) report_and_exit("Nem kaphato kulcs!");

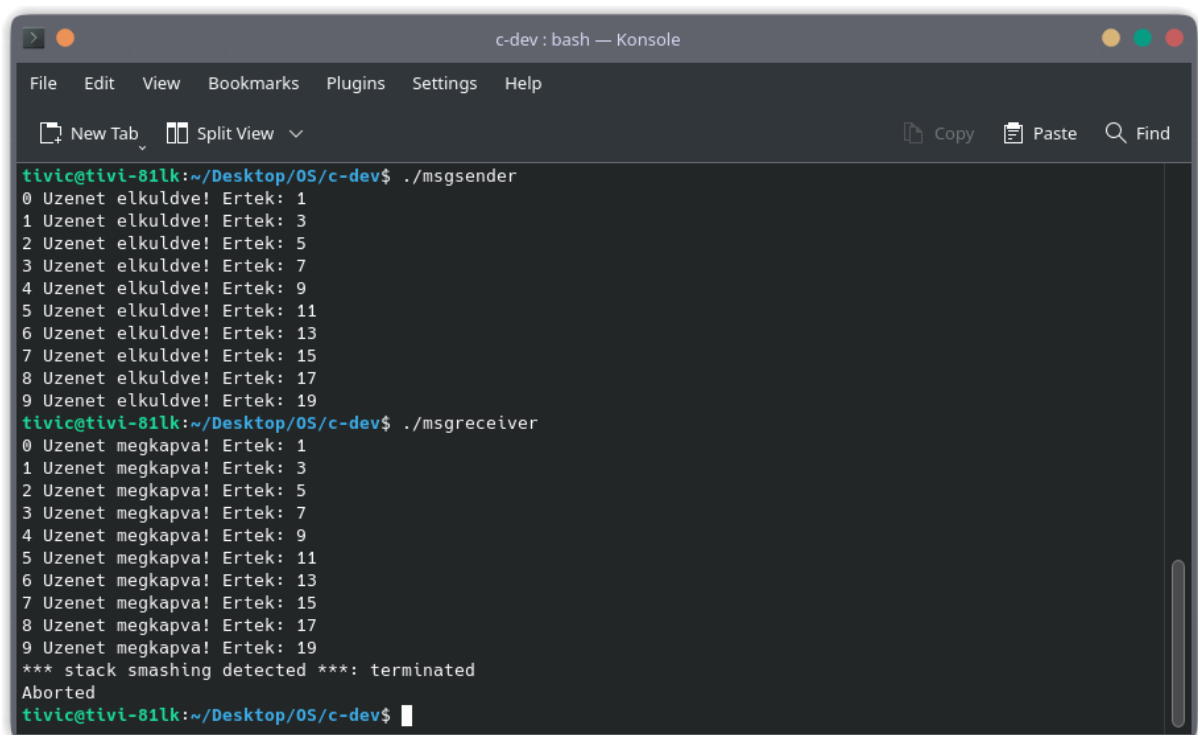
    int qid = msgget(key, 0666 | IPC_CREAT); /* eleri a qid-t, ha mar elkeszítve */
    if (qid < 0) report_and_exit("Nincs hozzaferes a sorhoz!");

    int i;
    for (i = 0; i < MsgCount; i++) {
        queuedMessage msg;
        if (msgrcv(qid, &msg, sizeof(msg), 0, MSG_NOERROR | IPC_NOWAIT) < 0)
            puts("Msgreceive problema!");
        printf("%i Uzenet megkapva! Ertek: %i\n", (int) msg.id, (int) msg.value);
        sleep(1);
    }

    /** uzenetsor eltavolitasa **/
    if (msgctl(qid, IPC_RMID, NULL) < 0) /* NULL = 'no flags' */
        report_and_exit("Problema az uzenetsor eltavolitasaval!");

    return 0;
}
```

A futtatás eredménye:



```
c-dev : bash — Konsole
File Edit View Bookmarks Plugins Settings Help
New Tab Split View Copy Paste Find
tivic@tivi-81lk:~/Desktop/OS/c-dev$ ./msgsender
0 Uzenet elkuldve! Ertek: 1
1 Uzenet elkuldve! Ertek: 3
2 Uzenet elkuldve! Ertek: 5
3 Uzenet elkuldve! Ertek: 7
4 Uzenet elkuldve! Ertek: 9
5 Uzenet elkuldve! Ertek: 11
6 Uzenet elkuldve! Ertek: 13
7 Uzenet elkuldve! Ertek: 15
8 Uzenet elkuldve! Ertek: 17
9 Uzenet elkuldve! Ertek: 19
tivic@tivi-81lk:~/Desktop/OS/c-dev$ ./msgreceiver
0 Uzenet megkapva! Ertek: 1
1 Uzenet megkapva! Ertek: 3
2 Uzenet megkapva! Ertek: 5
3 Uzenet megkapva! Ertek: 7
4 Uzenet megkapva! Ertek: 9
5 Uzenet megkapva! Ertek: 11
6 Uzenet megkapva! Ertek: 13
7 Uzenet megkapva! Ertek: 15
8 Uzenet megkapva! Ertek: 17
9 Uzenet megkapva! Ertek: 19
*** stack smashing detected ***: terminated
Aborted
tivic@tivi-81lk:~/Desktop/OS/c-dev$
```

Ennél a képnél az látható, hogy először a futtató programot indítottam el, ahol ki is írja az elküldött értékeket, az alatta lévő pedig a fogadó program futása látható, ahol a kapott értékek és az üzenet sorszámát is kiírja. Miután megkapta az összes üzenetet, törli az üzenet sort és kilép.